

The PCA (Principal Components Analysis) Program

Matthew Marcus

August, 2002

I. Introduction

PCA is a method for analyzing a set of spectra to see if they can be represented as linear combinations of a smaller number of component spectra. This program implements PCA as described by Ressler (Environ. Sci. Technol. **2000**, 34,950-958). The notation in the program is as used in that paper.

II. Main screen

The program starts out by asking for unknowns, one at a time. By default, files with extension *e* or *b* are shown in the file dialog. To end, hit Cancel in the file-open dialog box. After that, you see a screen which looks like that in Figure 1, except for the red labels.

This screen shot was taken with a demonstration dataset, each file of which was made by taking linear combinations of the same three references and then adding noise. This is therefore a highly-idealized case in which PCA will tell us what we already know, that there are three components.

At the upper left is the list of paths you selected. To the right of this is a set of checkboxes wired like radio buttons so you can only click one at a time. This selects which unknown is plotted, along with its reconstruction from the chosen set of components and the residual thereof. These data are shown on the graph to the right in white for the input data, green for the reconstruction and red for the residual.

Above the graph are two ‘badness-of-fit’ indicators. One of these is for the individual file shown in the graph, and the other is the average over the whole set. This quantity is defined as $\sum (y_i - y_i^{fit})^2 / \sum y_i^2$, with the index ranging over the points in an individual curve or the whole set.

The component list to the lower right shows the breakdown into the abstract components. Note that these components are at best linear combinations of spectra corresponding to species in the unknowns and often don’t look much like EXAFS. The first column in the list is the eigenvalues, whose squares represent the contribution to the

data made by that particular component. Next is the indicator (*IND*) value of Malinowski, a measure of the usefulness of adding in another component. According to the semi-empirical theory of errors in PCA, the last useful component is the one at which *IND* is a minimum. The next column is a set of checkboxes which let you add selected components to the fit. If you add all of them, as is the case when the program starts, the fit is perfect and perfectly meaningless. Here, we've added the first three. One usually adds them in order of decreasing eigenvalue. Next is another column of checkboxes allowing you to select one component to be plotted on the graph at lower right. This component is weighted by the eigenvalue, so insignificant components come out small. The contribution of each weighted component to the data being plotted in the upper graph is given by the column of numbers at the right edge of the components list complex. If a component is not selected for inclusion, its contribution to the whole is zero, regardless of the number in the contributions column. You can save each individual component to a file (default extension *cmp*). If you do a linear least-squares fit to one of the data files using these components as references, you will get the coefficients shown in the contributions column. Thus, in the present example, the first data file (10-10-80n.b) fits to $-0.162*comp1+0.495*comp2-0.562*comp3$, where *comp_i* is the *i*th component.

It should be noted that the method rarely works as well on real data as it does on this simulated data. Don't expect the residual to look like pure white noise even when you add the right number of components. In this example, the fourth and higher components all look like pure noise, while there's obvious signal in the first three (not shown). For real data, it's not so obvious and one must look to the *IND* value and prior knowledge about what's reasonable for the system.

III. Target transformation

The PCA fit gives no indication of what the components actually represent. One way of finding out is to use the target transformation. This procedure takes a reference data file and removes from it everything which doesn't look like something found in the unknowns. Thus, if the unknowns can be represented as mixtures containing the reference being tested, the target transformation will leave the reference spectrum

unaffected. Otherwise, the output won't look like the input. The screen in which this test is done is shown in Figure 2.

In this case, we have tested a reference which was not one of the three from which the simulated data were made. The *SPOIL* value (a measure of how much the target transformation disagrees with the input) is much greater than the 0-3 one expects to see for a reference which is really represented in the data, and the reconstruction looks nothing like the original. In most real cases, it's not so obvious that a reference doesn't belong, which is why the *SPOIL* value is computed. There are buttons which allow you to save the transformed file (default extension `trg`) and to read in another candidate reference.

It is assumed that the test reference data covers the same or almost the same range as the data from the unknowns. There is a method (iterative target factor analysis) for doing the target test which lets one violate this assumption, but this program doesn't do it.

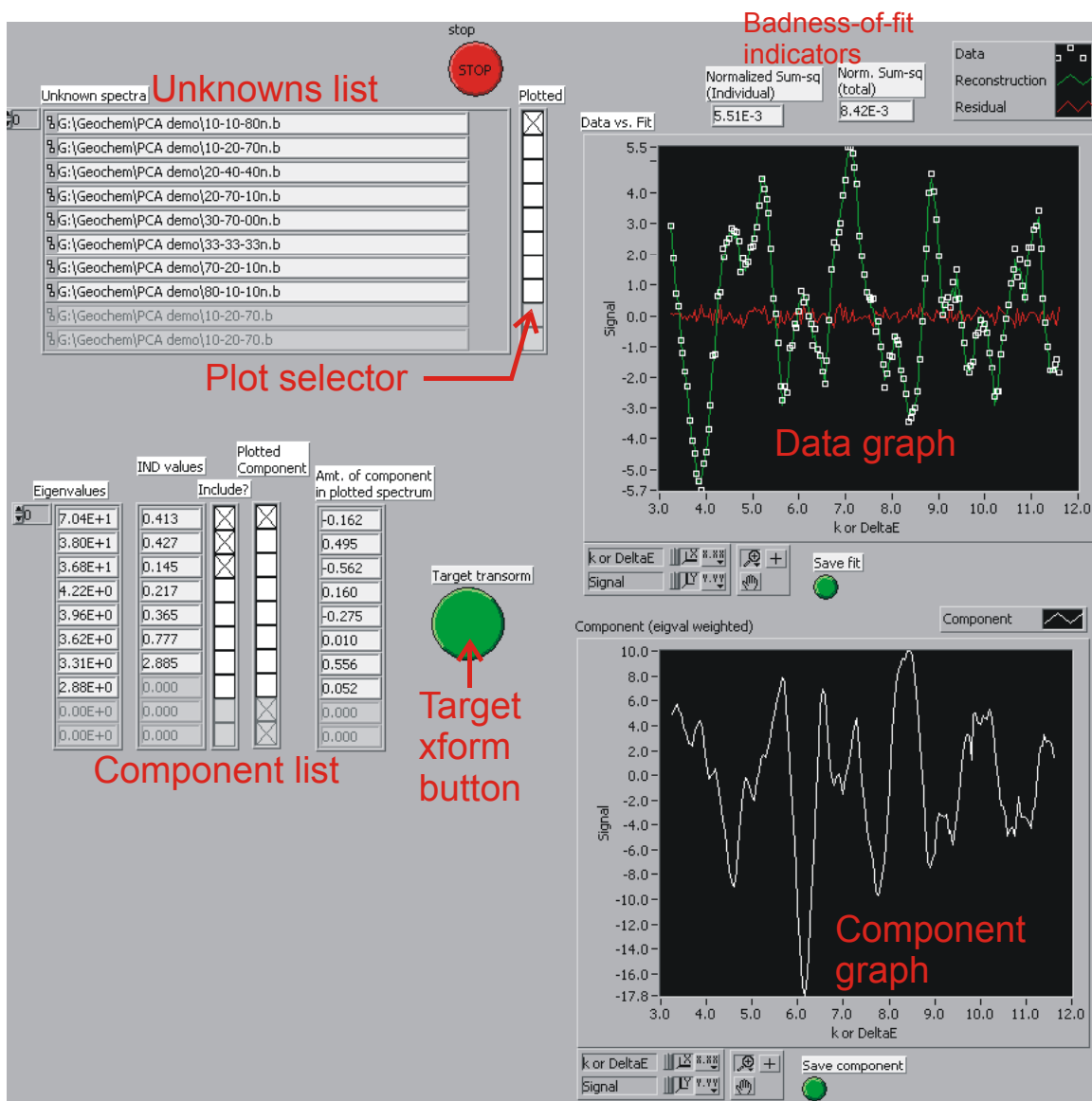


Figure 1. The main PCA screen, for a demonstration set of data. The fit has been limited to the first three components, the first one of which is shown in the component graph. The first data file is plotted along with a fit and the resulting residual.

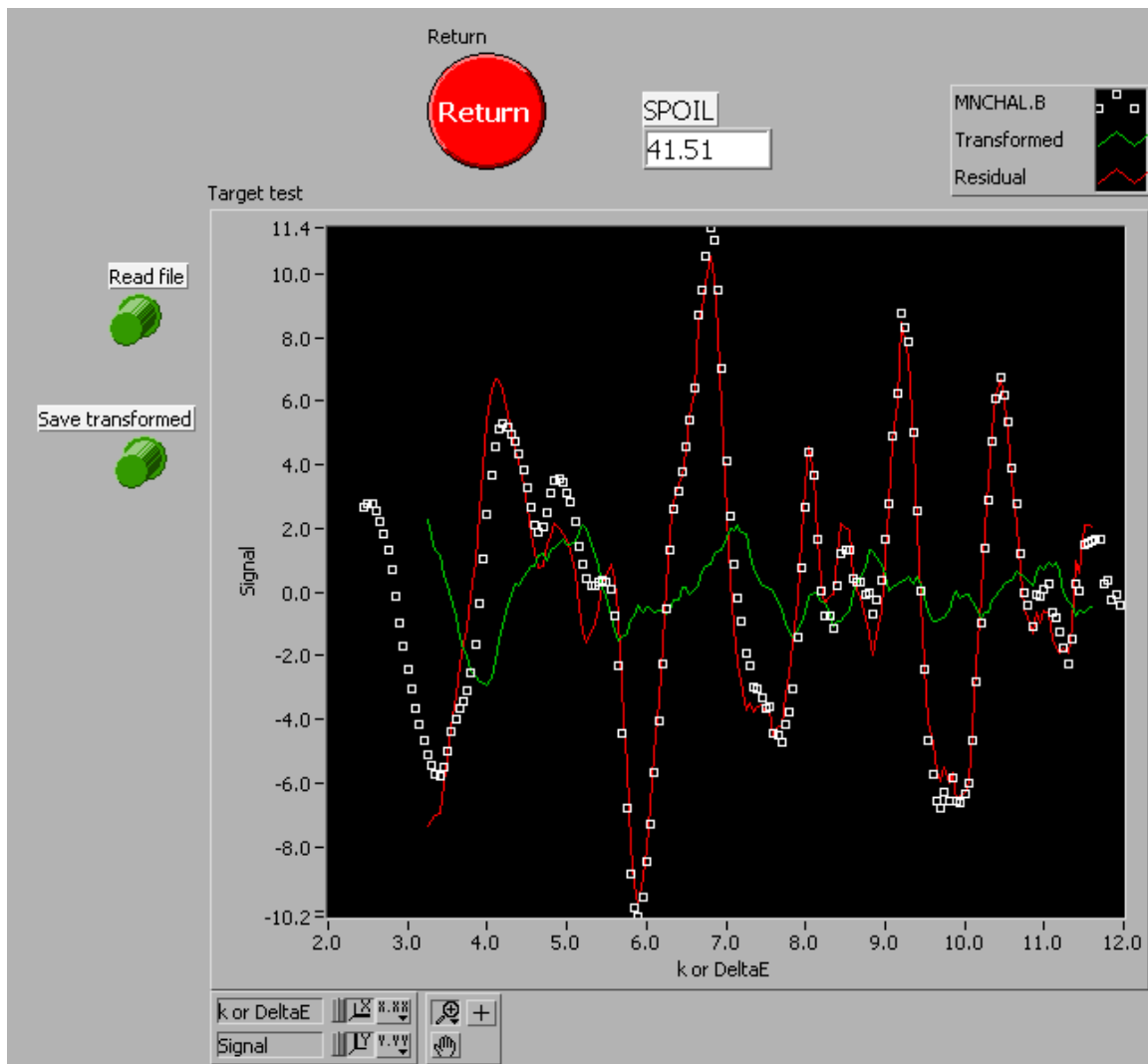


Figure 2. The target-transformation screen. The reference being tested is not one of the ones from which the demonstration data set was created. Therefore, the SPOIL value is high and the transformed data don't resemble the input.